Intel® Universal Plug and Play Software Development Kit V1.0 for Linux*

Technology Brief

Written by:
Daniel Baumberger, Intel Architecture Labs

Revision 1.1

Intel Corporation
5200 N. E. Elam Young Parkway
Hillsboro, Oregon 97124-6497

Revision History

Revision	Date	Description
Rev 1.0	August 31, 2000	Final Draft
Rev 1.1	September 14, 2000	Revised format and content for Web site distribution

1 Introduction

Universal Plug and Play (UPnP) allows automatic discovery and control of services available on the network from other devices without user intervention. Devices that act as servers can advertise their services to clients. Client systems, known as control points, can search for specific services on the network. When they find the devices with the desired services, the control points can retrieve detailed descriptions of the devices and services and interact from that point on.

This technical brief describes the high-level architecture of the Universal Plug and Play Software Development Kit (SDK) V1.0 for Linux*. This document describes the main pieces in the implementation and what is included in the UPnP SDK.

1.1 References

Reference Number	Document	Revision
1	*Simple Object Access Protocol (SOAP)*, D. Box, DevelopMentor, Inc.	IETF Draft revision November 1999
2	*Universal Plug and Play (UPnP) Device Architecture*, Microsoft Corporation	Revision 1.0, June 8 2000
3	*Universal Plug and Play V1.0 for Linux API Reference*, Intel® Corporation	Revision 1.0.0, August 31, 2000

2 UPnP Overview

A brief description of UPnP is given in this section. For more information, see [2].

UPnP includes five basic phases:

1. *Discovery*. In this first phase, control points search for devices and services. Similarly, devices multicast announcements of services they offer.
2. *Description*. Once a control point finds an interesting device or service, it requests from the device for a complete description of the device, its component devices, and services.
3. *Control*. This phase allows control points to enact changes in the state of a device thus causing the device to perform some action.
4. *Eventing*. The eventing phase allows control points to keep in synch with the state of services in which it is interested. Control points subscribe to the event server for a particular service and receive event notifications when that service's state changes.
5. *Presentation*. The presentation phase allows a device to host a document, written in standard HTML, which can be a user interface for that device.

Each of these basic phases is briefly described in the following sections.

2.1 Discovery

In the discovery phase control points find devices and services, and devices announce their presence to control points using the Simple Service Discovery Protocol (SSDP). SSDP uses a variant of HTTP that operates over multicast UDP for broadcasts and another variant of HTTP that operates over unicast UDP for replies.

A device may consist of other devices, each with its own services. Devices are identified both by type and by a unique identifier. Services are identified by their type.

To search for devices or services on the network, control points use the HTTP M-SEARCH command multicast to the address 239.255.255.250:1900 over UDP. Any device on the network that matches the criteria the control point is searching for issues a unicast UDP reply that includes the URL to its description document (see section 2.2). If a control point receives one or more acceptable replies, it moves into the description phase.

Devices don't have to wait for a control point to search for their services. They can advertise their device availability by means of the GENA (General Event Notification Architecture) NOTIFY command on the 239.255.255.250:1900 multicast address. When control points see this NOTIFY multicast, they can request the device's description document. Devices must send a notification when their services will no longer be available.

2.2 Description

When a control point locates a service it wants to know more about, it requests the description document. The description is an XML document describing the device, including:

- Manufacturer information, version, and so on
- Any URLs to icons that can be used for the device
- A list of embedded devices
- A list of services supported by the device

For more information on the format of the description document, see [2].

The control point requests the description document using HTTP over TCP. The control point performs a standard HTTP GET command (similar to retrieving a Web page). On the server side, the device runs a standard HTTP server—either a full Web server such as Apache or a mini-server. Many of the elements in the description document are URLs. Those elements are also retrieved using HTTP/TCP.

2.3 Control

Once a control point discovers a device and retrieves its description document, it may want to control one or more of the services contained in the device. The Simple Object Access Protocol (SOAP) allows a control point to query or change elements in a service's state table. SOAP uses the PUT HTTP command transported over TCP.

SOAP uses XML to specify what actions to take. The control point creates the XML document and posts it to the control URL for the service, as specified in the description document. The control point can request current values and make changes to the service's state table.

On the server side, the control server waits for control requests. The control server is an HTTP-like server implementing the SOAP protocol. A device can operate more than one control server depending on the combination of services provided by the device.

2.4 Eventing

After a control point discovers a device and retrieves its description, it can stay informed of the state of a service offered by that device. Interested control points subscribe to the device's event notification service URL found in the description document for the particular service. An event notification is sent to the control point any time the state of the service changes, even if the control point causes the change.

Subscribe and unsubscribe requests use HTTP/TCP to connect to the event URL contained in the description document for the service. The control point specifies an URL where event notifications are made during subscription. Events arrive by means of HTTP/TCP to the URL registered with the service. The event notification includes a small XML document that describes the actual event, such as a change in the state table for the service.

On the server side, an event server waits for subscribe and unsubscribe requests. The event server is an HTTP-like server implementing the General Event Notification Architecture protocol (GENA). A device may have to operate more than one event server depending on the combination of services provided by the device.

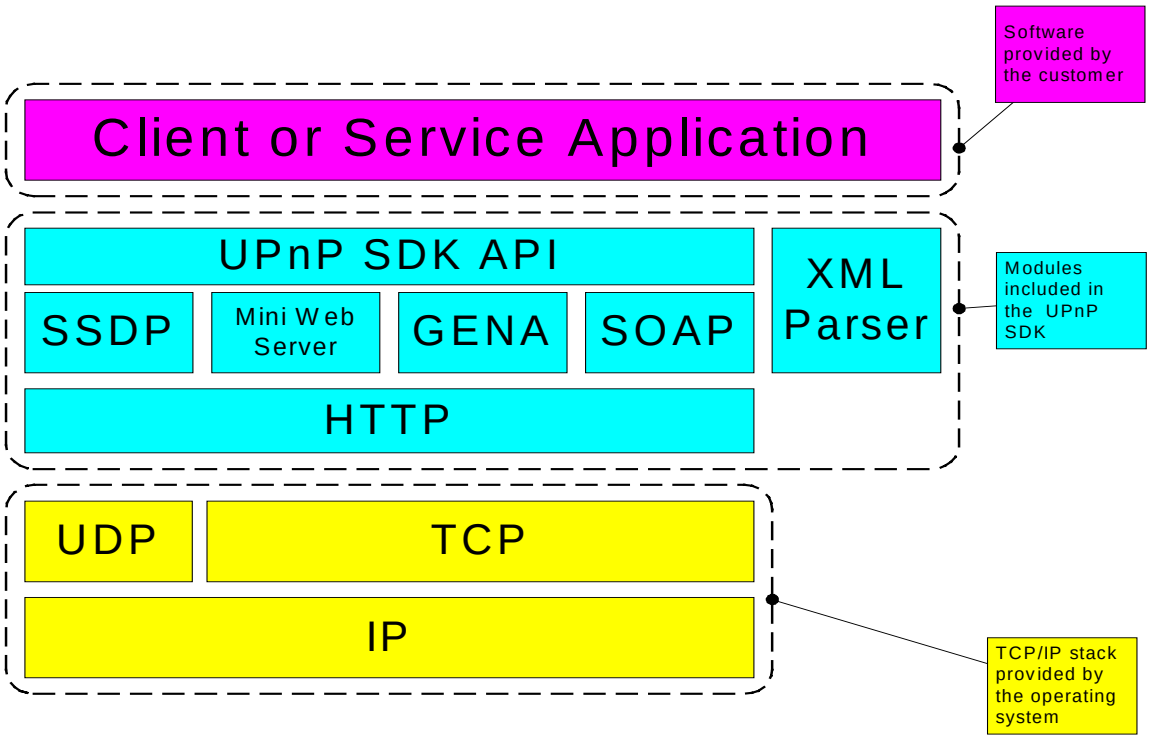
2.5 Presentation

For devices that need or support user interaction, in the presentation phase a control point can download an HTML document that represents the user interface for the device. This is a standard HTML document that can provide a means of control or status display.

The protocol for retrieving the presentation document, as with the description document, is HTTP over TCP. The control point can use the presentation URL contained in the description document to request the presentation document. Not all devices have a presentation document nor are all control points able to display a presentation document containing complex HTML objects such as frames, embedded Java* applets, and so on.

3 Architecture

The following diagram shows the architecture of the UPnP SDK for Linux:



The following sections describe each part of the diagram. For more information on any of the protocols, consult the appropriate specification or draft specification.

3.1 Client or Service Application

The customer provides client or server application software to run on top of the UPnP SDK. The client or server application implements the functionality of a specific service. For example, for a gateway service, the server software implements the "Enable Internet" functionality that the control point software can control using UPnP. A sample service and control point application is included as part of the UPnP SDK.

3.2 UPnP SDK API

The UPnP SDK API abstracts the details of the core UPnP protocols away from the control point or service application and gives applications access to the functionality in a unified interface. This frees the developers from concern about the details of the SSDP, GENA, and SOAP protocols in their code.

For information about the API, see [3].

3.3 SSDP

The SSDP module implements the Simple Service Discovery Protocol, providing the discovery phase of UPnP. This module allows control points to send multicast searches for services and devices on the network and receives the replies to those searches. It also notifies them when new services are announced on the network.

This module also allows servers to multicast announcements of their services to the network.

3.4 Mini Web Server

The Mini Web Server module handles the standard HTTP GET requests. Many UPnP elements are requested using this basic HTTP service. This module manages the locations of documents that are available using the GET command and implements the actual streaming of the data using the HTTP protocol.

3.5 GENA

The GENA module implements the General Event Notification Architecture, providing the eventing phase of UPnP. Control points use this module to subscribe or unsubscribe to services of interest. Service applications receive subscribe and unsubscribe notifications from this module and generate the appropriate events.

3.6 SOAP

The SOAP module implements the Simple Object Access Protocol, providing the control phase of UPnP. Control points use this module to generate the appropriate XML documents to retrieve or change the state tables of a service. The server uses this module to decode the control requests and generate the correct responses.

3.7 HTTP

The HTTP layer provides common functionality for GENA, SOAP, SSDP, and the mini-Web server. This layer accepts all HTTP connections, determines which HTTP request is coming in, parses all the HTTP headers, and transfers the connection over to the appropriate protocol for processing. The first line of the HTTP header contains the request. The HTTP layer minimally handles these HTTP commands:

- GET. Control points use the GET command to retrieve the description document and any of its sub-elements including the presentation document, the service description documents, and the icons associated with the device.
- POST/M-POST. A SOAP command is in the form of a POST or an M-POST command. All POST commands are transferred to the SOAP module for further processing.
- SUBSCRIBE. SUBSCRIBE commands are transferred to the GENA module for further processing. Control points use SUBSCRIBE requests to subscribe or renew a subscription to event notifications for a particular service.

- UNSUBSCRIBE. UNSUBSCRIBE commands are transferred to the GENA module for further processing. Control points use UNSUBSCRIBE commands to notify a server that they are no longer interested in receiving event notifications.
- NOTIFY. NOTIFY commands are transferred to the GENA module for further processing. NOTIFY commands are event notifications sent from servers to control points containing a description of the event.

3.8 XML Parser

XML is used extensively in UPnP. The description documents are XML documents. GENA uses XML to describe change in a service's state. SOAP uses XML to format requests and responses. The UPnP SDK contains an XML parser used both by the core UPnP protocols and by the client or server software.

The interface to the XML Parser uses a subset of the Document Object Model (DOM) Level 1 recommendation written by the World Wide Web Consortium (W3C). The UPnP SDK provides both C and C++ interfaces. See http://w3c.org/DOM/ for more information on DOM.

3.9 UDP/TCP/IP

The UPnP SDK assumes the TCP/IP stack is provided by the operating system. Although the TCP/IP stack is not part of the UPnP SDK, it is included in the diagram to illustrate the protocol used for the core UPnP protocols.

4 Sample Operation

4.1 Client Sample

The UPnP SDK for Linux includes a sample control point application. This application works only with the sample device application (see section 4.2). However, this sample shows how a control point application may be written using the UPnP SDK for Linux.

For details on building and running the sample, see the README file in the sample directory of the UPnP SDK distribution.

4.2 Server Sample

The UPnP SDK for Linux contains a sample device application. It emulates a TV device and allows changing of the channels, volume, and some other settings. The device accepts commands from a Linux system using included control point sample (see section 4.1) or from a system running Microsoft Windows Millennium Edition.

For details on building and running the sample device, see the README file in the sample directory of the UPnP SDK distribution.

5 Operating Environment Dependencies

The Universal Plug and Play Software Development Kit for Linux is written using standard IEEE Portable Operating System Interfaces (POSIX). For details on dependencies on other system libraries and packages, see the README file included with the UPnP SDK for Linux.